Quo vadis, crypto validation?

Apostol Vassilev

March 11, 2019

Computer Security Division, NIST

Background

What is FIPS 140?

FIPS 140 is a Federal Information Processing Standard

- specifies the security requirements for cryptographic modules
- based on the Computer Security Act of 1987
- references all NIST-approved crypto primitives Annex A-D
- mandated by FISMA 2002

NIST established the Cryptographic Module Validation Program (CMVP)

- people often interchangeably refer to the standard and the program as FIPS 140
- crypto implementations must be validated to be used by Federal Agencies

Why validate cryptography?

Interoperability and security are the primary reasons

FACT:
In modern commercial cryptography the algorithms are known.

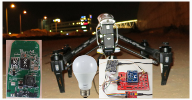
So,

> the security hinges on the secrecy of keys and internal state
>
> > e.g., the **black box assumption** in theoretical cryptography
> > ... but practice comes with different challenges

Cryptography is affected by implementation vulnerabilities

Attacks exploit differences between ideal and real implementations:

“ZigBee Chain reaction” (2017)



key ex-filtration via side-channel leaks

Meltdown&Specter (2017) + Foreshadow (2018)

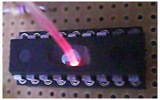
, 

user/VM separation compromise due to speculative execution

Heartbleed (2014)

key ex-filtration due to buffer over-read

Belcore attack (1997)



induced RSA-CRT computation error leading to modulus factorization

Cryptography is difficult to understand and implement correctly

FACT:
"Usability is often neglected in cryptographic resources such as standards and libraries, resulting in complex solutions that provide little assistance to developers in making secure choices"

Major complaint:

complexity of the language in the standards

developers could benefit from more explanations of motivation - the "why" behind cryptographic choices.


Challenge: develop standards for threshold cryptography that are accessible by a (more) general audience?

"We make it a big deal in the company": Security Mindsets in Organizations that Develop Cryptographic Products,

Haney, Theofanos, Acar, Prettyman.

Some security requirements are not testable

Proposed draft of FIPS 186-5:

- The standards defines the following acronyms:

  DRBG - Deterministic Random Bit Generator, specified in SP 800-90A Rev1.

  RBG - Random Bit Generator

- Furthermore, Algorithm B.3.3 (Random Probable Primes), in Step 4, Generate *p*, asks:

  4.2 Obtain a string *p* of (nlen/2)-bits from an RBG that supports the *security_strength*.

How is this prime generation method tested?

There are no approved RBG's, hence CAVP only verifies the primality of *p*

What's the problem here?

Infineon's prime generation method (ROCA vulnerability 

) would have passed!

Fortunately, there is an easy fix: replace RBG with DRBG in 4.2.

Some crypto algorithms are brittle

BRITTLE (Dictionary.com):

- having hardness and rigidity but little tensile strength; breaking readily with a comparatively smooth fracture, as glass.
- easily damaged or destroyed; fragile; frail.

Example:

AES-GCM - an authenticated encryption block cipher, NIST standard - SP 800-38D.

very efficient, widely used

BUT...

AES-GCM: Key/IV uniqueness critical for security

SP 800-38D: In practice, this requirement is almost as important as the secrecy of the key

- How clear is this requirement for developers?
  - None of the major cryptographic platform API's paid attention, for many years (10+)
  - Unsuspecting developers assumed the risk of getting it wrong
  - Only recently PKCS#11 moved to provide default safe handling of IV's
- It is difficult to come up with a test for uniqueness of key/IV combinations
  - Some protocol specifications (TLS, IPSec) handle it well - at the protocol level
  - Other applications are much harder to handle
  - CMVP relies on naked-eye code inspection

What's the lesson here?:

In standards avoid critical security requirements that are easy to get wrong and for which there is no objective machine test

Are these challenges important to pursue? Why do we care about them?

The big picture in cybersecurity

Key findings

The impact is felt across the whole business:

from your legal team, embroiled in litigation,

to your frontline employees, who can't access the tools they need to do their jobs.

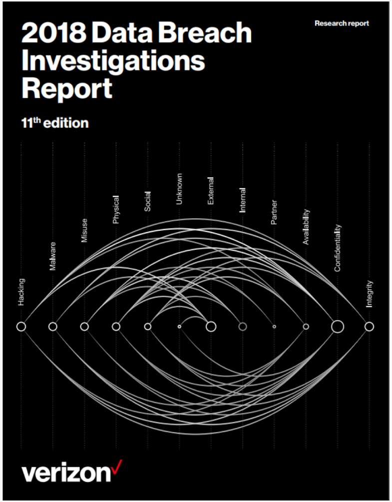
2017 – a "banner" year of cybersecurity failures

worse than the prior - a troubling multiyear trend

Key recommendations

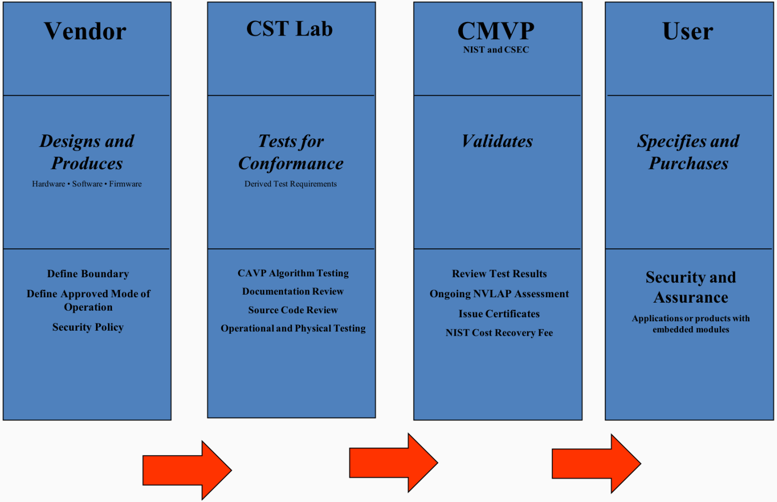
The two most relevant for us:

- encrypt sensitive data
- patch promptly

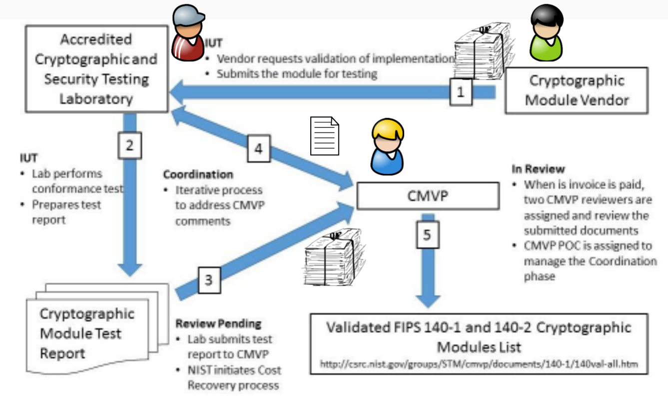


How does the NIST crypto module validation program work?

Traditional CMVP Testing and Validation



Current CMVP Process



 Process relies entirely on human actors and human-readable artifacts (English essays?).

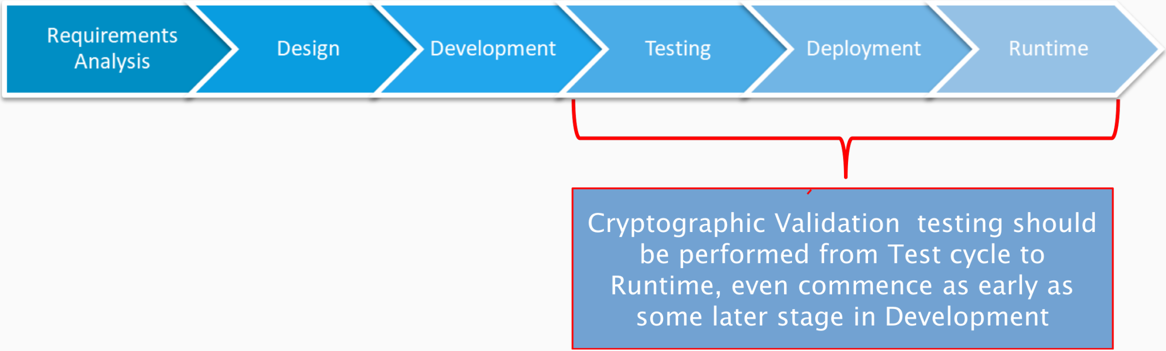
Is this validation model adequate for the challenges facing us?

George Orwell: To see what is in front of one's nose needs a constant struggle

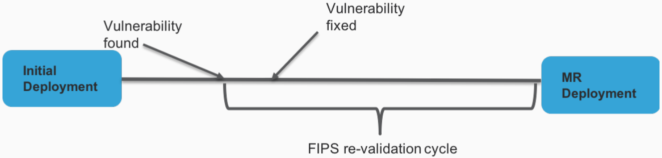
Key Findings of Industry/Government WG

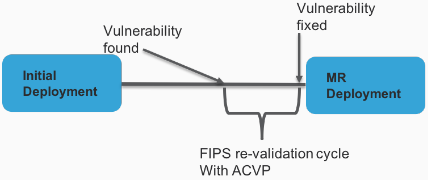
- Long review cycles
  - well beyond industry product development cycles
  - costly and rigid
  - slows adoption of latest technology
- Subjective reviews
  - different reviewers render different judgements on same report
  - humans are susceptible to manipulation by the style of report writing
- Shallow testing of security requirements
  - software testing not covered well
  - hardware security testing is subjective
- Inability to get FIPS 140-2 compliance assurance on platforms of interest
  - tested configurations do not match real platforms

Typical Industrial Product Development Lifecycle: Current and Desired



Industrial Product Maintenance Lifecycle: Current and Desired





Can these problems be solved within the existing envelope?

Some tweaks of the current program offer minimal improvement. However, this model cannot scale up so that

- the latency of testing would decrease,
- the latency of review would decrease,
- the objectivity of reviews would increase
- the depth of testing would improve significantly,
- the costs would decrease,

all simultaneously.

Out-of-the box approaches are needed

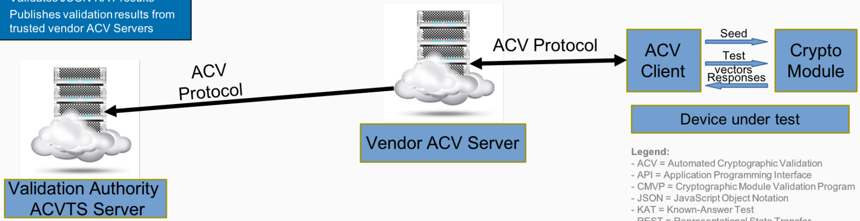
Our approach to the solution

Building a new crypto validation program

Validation Authority Server:
- Web hosted service w/ REST API
- Registers ACV Servers
- Generates JSON KAT vectors
- Validates JSON KAT results
- Publishes validation results from trusted vendor ACV Servers

ACV Proxy/Server:
- Web hosted service
- Interacts with NIST ACV Server to obtain JSON KAT data
- Optionally generates JSON test vectors
- Optionally performs results verification
- Reports JSON KAT results to NIST ACV Server

ACV Client:
- Integrated into Device under test
- May convert JSON test vectors to format acceptable by crypto module under test
- Returns KAT answers to ACV server in JSON format

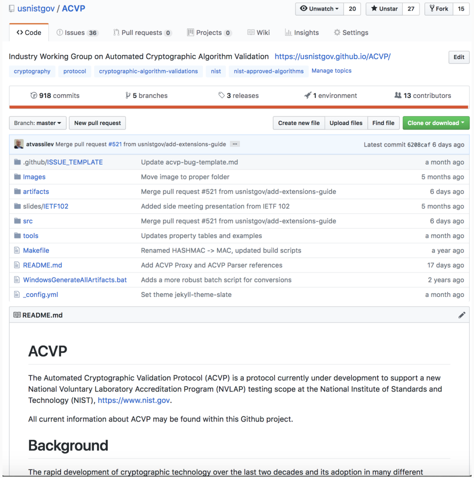


Legend:
- ACV = Automated Cryptographic Validation
- API = Application Programming Interface
- CMVP = Cryptographic Module Validation Program
- JSON = JavaScript Object Notation
- KAT = Known-Answer Test
- REST = Representational State Transfer
- ACVTS = Automated Crypto Validation Testing Service

Computer-based testing and validation 

Where are we now?

Algorithm testing - provides base automation infrastructure



Targeting late Q1, 2019 for deployment

- all CAVS algorithms are implemented and testable
  - the server supports even more algorithms
  - improvements of testing methodology for some algorithms
  - currently stress-testing, preparing for deployment
- ACVP testing scope developed for **HB 150-17** and accepted by NVLAP. Open for accreditation, a few applications on the way.
- working on standardizing the protocol and testing methodology with IETF

Module testing - the Holy Grail of validations

Working on pilots with different technologies

Red Hat - software

Apple - software and hardware

Google - hardware

Amazon Web Services - cloud

Work independently, then collaborate

- by-weekly meetings with individual pilots
- monthly coordination meeting with all

Leverage the protocol and infrastructure established by ACVP

- Developing a schema and protocol for module test results submission
- Developing Machine (Deep) Learning/AI-based analysis of test reports
- Targeting mid-2020 for potential deployment

A bit about FIPS 140-2 security test requirements

FIPS 140-2: a (large) set of security assertions, vendor documentation requirements and test requirements that support them

- only a subset of all requirements apply to a particular module
- depending on the embodiment (hardware, software, hybrid) the applicable requirements vary a lot
- even within an embodiment, e.g., software, the test requirements very from one module to another
- depending on the assurance level (1-4), the test requirements vary a lot
- some requirements apply to all modules, others are conditional on the available functionality

    standalone vs. a set of dependent requirements

An example

AS04.05: (Levels 1, 2, 3, and 4) Documentation shall include a representation of the finite state (or equivalent) using a state transition diagram and/or state transition table

> TE04.05.08: The tester shall exercise the cryptographic module, causing it to enter each of its major states. For each state that has a distinct indicator, the tester shall attempt to observe the indicator while the module is in the state. If the expected indicator is not observed, or two or more such indicators are observed at the same time (indicating that the module is in more than one state at one time), this test fails.

Sample JSON test results

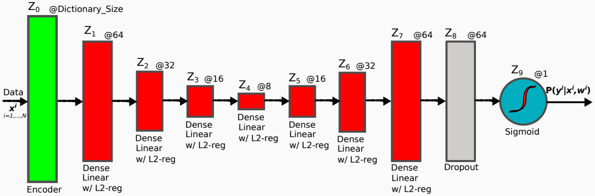

```
[
    {‘‘testEvaluations’’ : [

            ‘‘test’’ : ‘‘TE04.05.08’’,
            ‘‘type’’ : ‘‘dynamic’’,
            ‘‘result’’ : ‘‘pass’’,
            ‘‘assessment’’ : ‘‘pk11mode tests all the Module states as
 Loading the library takes the Module from Power off (1.x) to Power Up
   Test (1.B)
 Power Up Self Test (1.B) proceeds to Inactive (1.A) on success of thos
 C_Initialize takes the Module from Inactive (1.A) to Public
  Services (1.C)
 C_Login takes the Module from Public Services (1.C) to NSS User
  Services (2)
```

... continued: sample JSON results for testing module integrity checks

```
''log'' : ''debuggerTestLog''
''logText'' : ''catch-load libsoftokn3.so
-catch-load libnssdbm3.so
....
#breakpoint8 called
#breakpoint 8 reached with RSA signature check complete
-exec-continue
^running
*running,thread-id=\"1\"
:Copying library files from /lib64/ to /tmp/amvp_nss_16084
:mangling file libsoftokn3.so
:attempting to open FIPS token with mangled softokn3
&\"Detaching after fork from child process 16090.\n\"
Simple Test running, Expecting Failure
Simple Test: C_Initialized failed as expected with 0x00000030
            (CKR_DEVICE_ERROR)
```

BowTie - deep learning for sentiment prediction



Feedforward neural network

- combined with a polarity model of the semantic content
- trained model with 92% transfer accuracy
- capable of predicting sentiment of 50,000 texts with average length 200-words in $<$ 2 minutes
- Working with academics to refine the polarity model for the technical jargon used in assessments

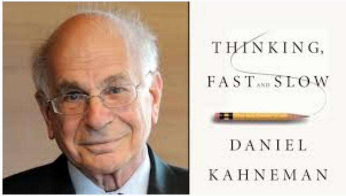
Is deep learning appropriate for validation of critical for security modules?

How can you trust a system that is not 100% accurate?

If a person was reviewing this she would have never made a mistake!

- people are confident they can perform cognitive tasks well

Homo sapiens and their brains



The sapiens are creatures of dual thinking, fast and slow, that shapes their perception and choice

- System 1 **(Fast)** operates automatically and quickly
- System 2 **(Slow)** allocates attention on effortful mental activities that demand it, including reasoning and complex computations

Although System 2 believes itself to be where the action is, the automatic System 1 is the hero.

Example

Activities attributed to System 1:

- Detect that one object is more distant than another.
- Orient to the source of a sudden sound.
- Complete the phrase "bread and . . ."
- Make a "disgust face" when shown a horrible picture.
- Understand simple sentences.

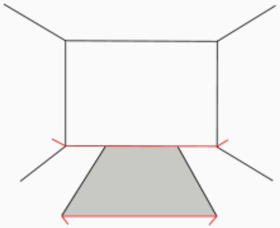
Activities attributed to System 2:

- Focus on the voice of a particular person in a crowded and noisy room.
- Count the occurrences of the letter 't' in a page of text.
- Tell someone your phone number.
- Fill out a tax form.
- Check the validity of a complex logical argument.

Interaction between the systems of our thinking

Some facts about the interaction between Systems 1 (Fast) and 2 (Slow):

- The division of labor between Systems **1** and **2** is efficient: **1** on auto-run, **2** in a low-effort mode
  - System **1** continuously generates suggestions for **2**: *impressions, intuitions, intentions*, and *feelings*.
  - If endorsed by System **2**,
    - *impressions* and *intuitions* turn into **beliefs**
    - *impulses* turn into **voluntary actions**
- But System **1** cannot be turned off and has biases/system errors that it is prone to make.



Which red line is longer?

Franz Carl Müller-Lyer, a German sociologist, 1889 (Wikipedia)

"We can be blind to the obvious, and we are also blind to our blindness."

Final thoughts

Standardization and validation of threshold cryptographic schemes are challenging

- Need well-understood and robust schemes
- Need well-written, testable standards

Important to select schemes that can be fully tested using machine-based approaches

- suitable for the new automated validation program - the only viable validation alternative

Questions?